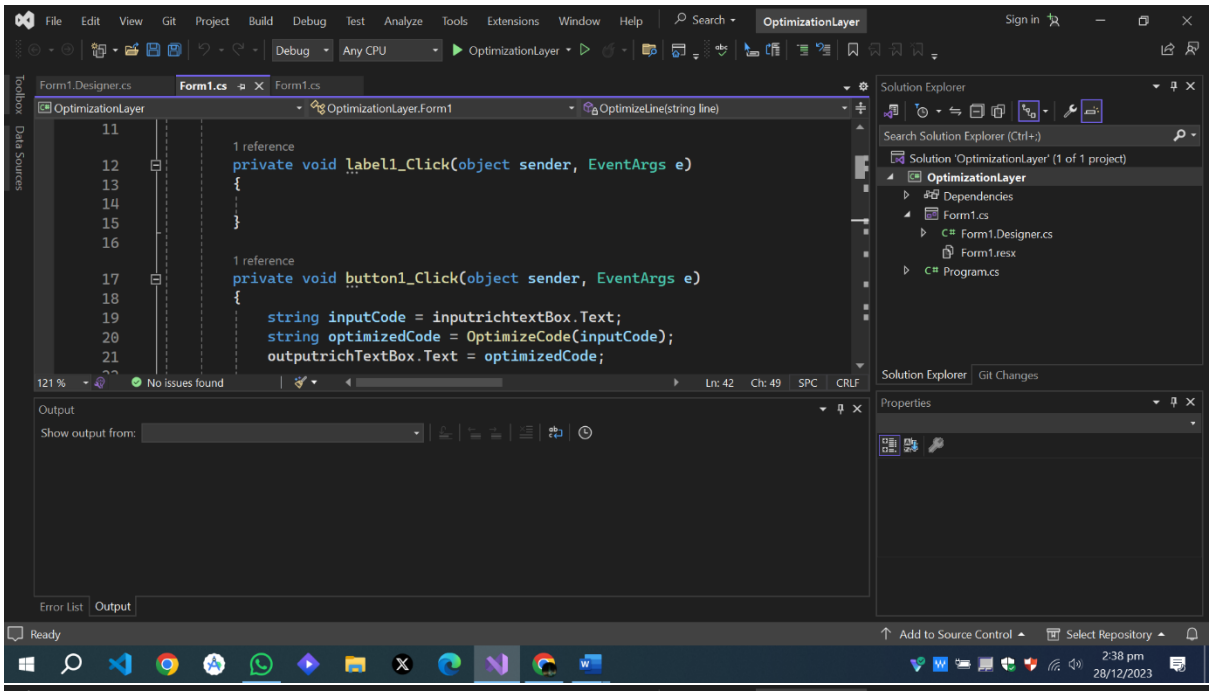
Q2) Give two functionalities along with screenshots.

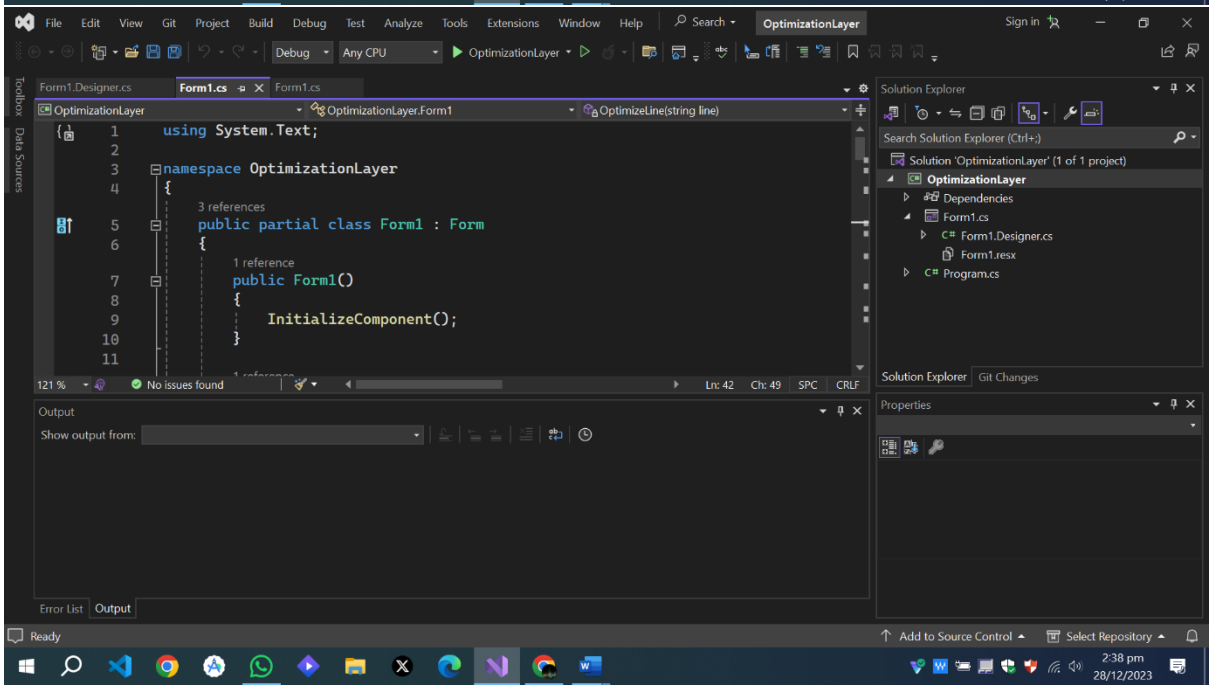
Constant Folding and Propagation:

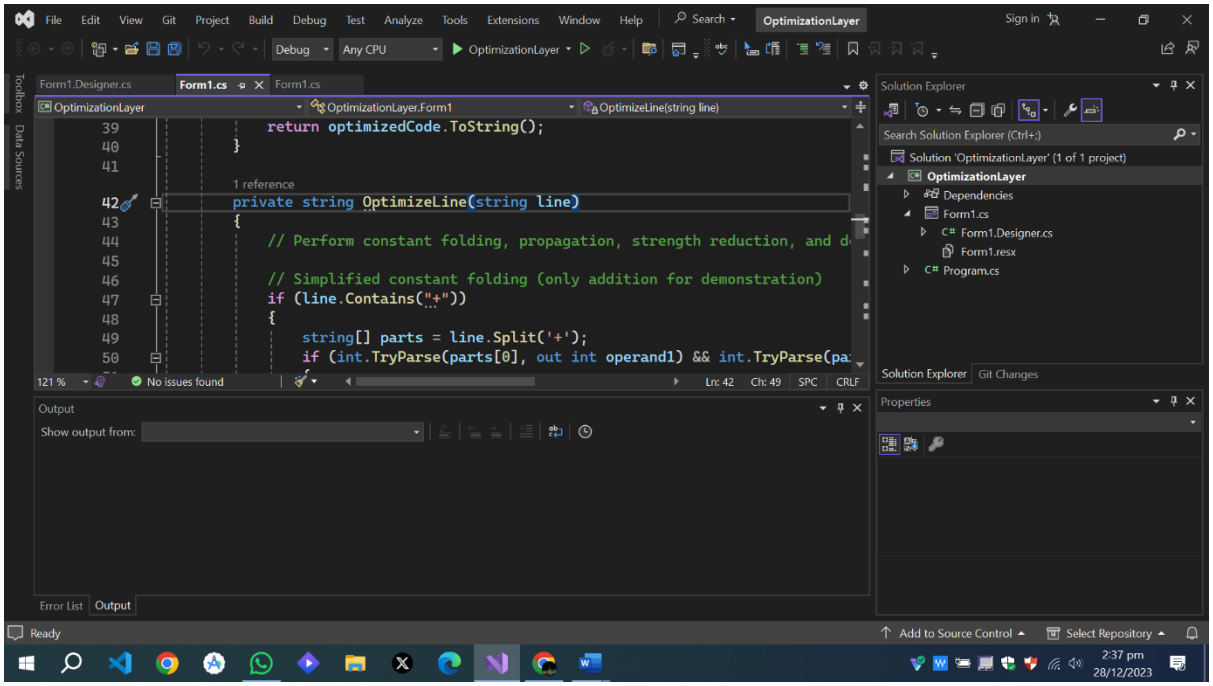
The analyzer identifies constant expressions within the input C# code and evaluates them at compile time.

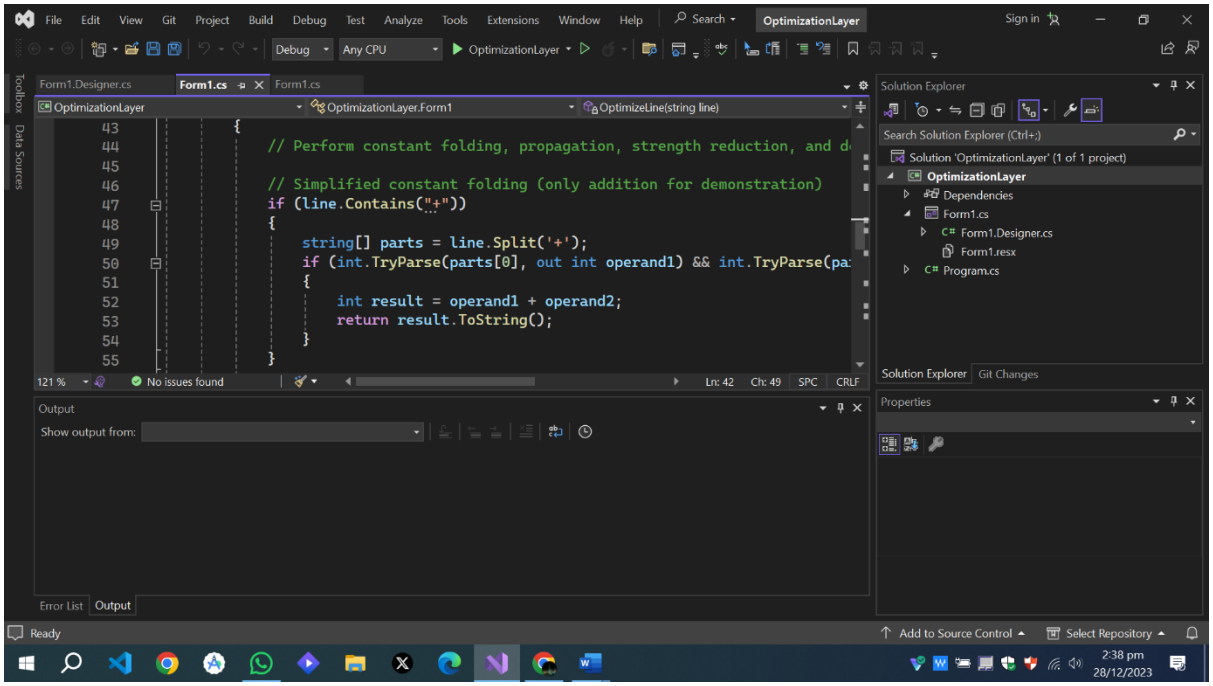
It replaces these constant expressions with their computed values where possible to reduce the need for runtime computations.

This functionality aims to improve the program's efficiency by minimizing the computational overhead associated with repetitive calculations of constant expressions.









Dead Code Elimination:

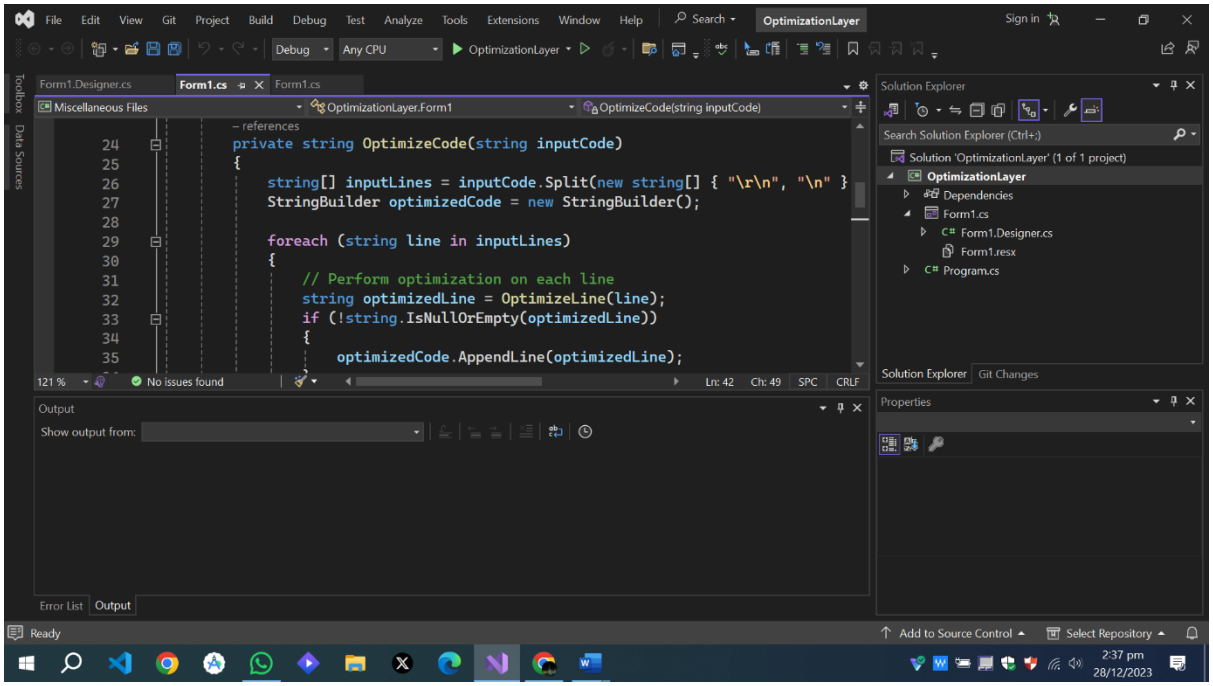
The analyzer identifies and eliminates code segments that do not impact the output of the program.

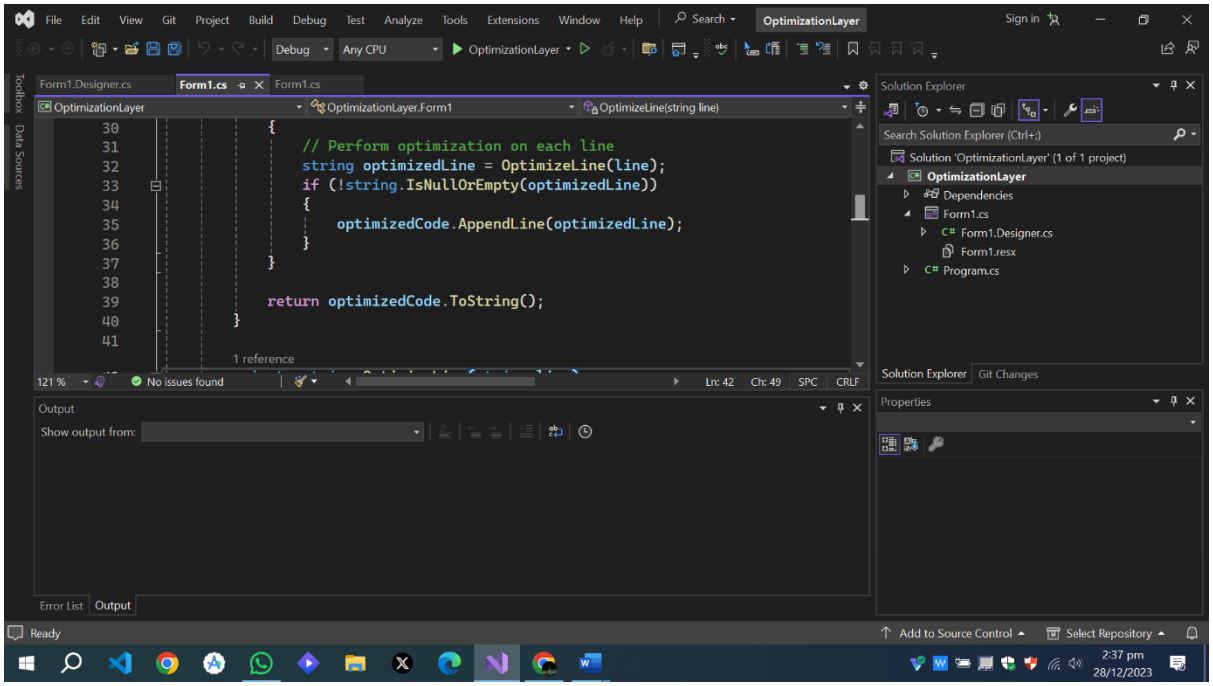
It helps reduce the size of the program by removing unused or redundant code, leading to more streamlined and efficient execution.

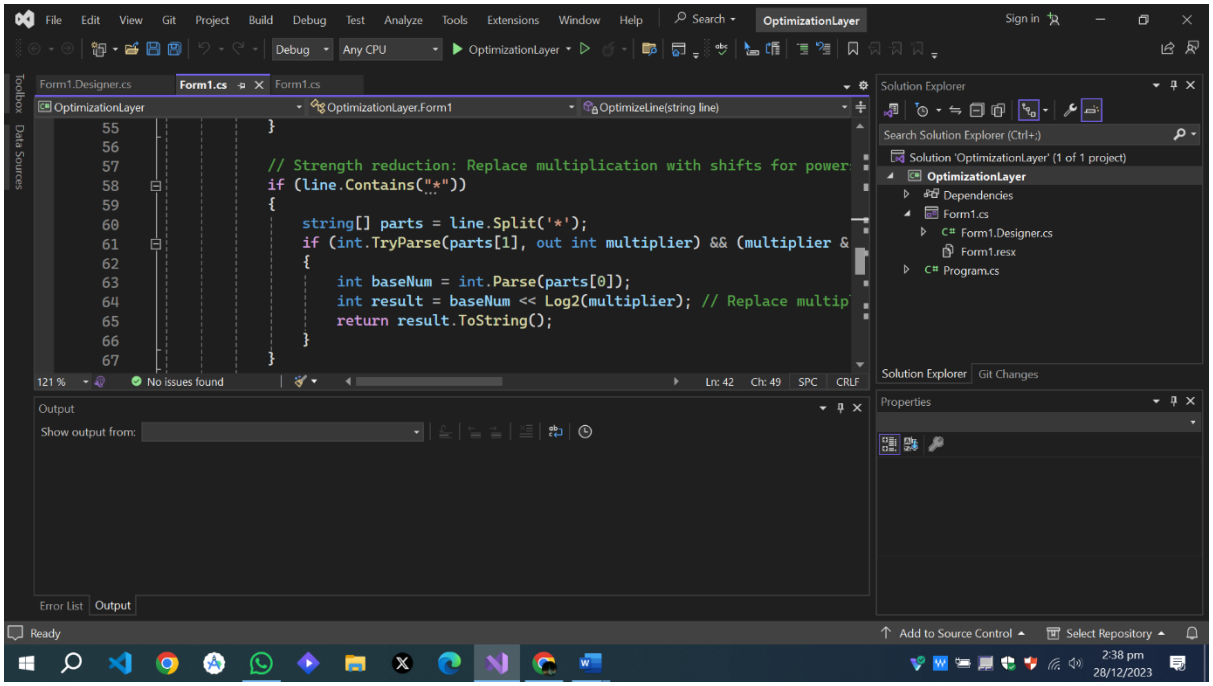
This functionality contributes to enhancing the program's performance and makes the codebase more manageable and maintainable by removing unnecessary clutter.

Strength Reduction:

The strength reduction functionality in the C# Optimization Analyzer identifies complex operations in the code and replaces them with simpler and more efficient alternatives, such as substituting multiplications with additions or bit shift operations. This optimization technique aims to improve code performance without altering the program's functionality, offering valuable educational insights for developers.




File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
OptimizationLayer
Sign in
Debug
Any CPU
OptimizationLayer
Form1.Designer.cs
Form1.cs
Form1.cs
OptimizationLayer
OptimizationLayer.Form1
OptimizeLine(string line)
{
// Perform optimization on each line
string optimizedLine = OptimizeLine(line);
if (!string.IsNullOrEmpty(optimizedLine))
{
optimizedCode.AppendLine(optimizedLine);
}
}
return optimizedCode.ToString();
}
1 reference
121 %
No issues found
Ln: 42
Ch: 49
SPC
CRLF
Output
Show output from:
Error List
Output
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'OptimizationLayer' (1 of 1 project)
OptimizationLayer
Dependencies
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Solution Explorer
Git Changes
Properties
Ready
Add to Source Control
Select Repository
2:37 pm
28/12/2023


File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help
Search
OptimizationLayer
Sign in
Debug
Any CPU
OptimizationLayer
Form1.Designer.cs
Form1.cs
Form1.cs
OptimizationLayer
OptimizationLayer.Form1
OptimizeLine(string line)
}
// Strength reduction: Replace multiplication with shifts for power
if (line.Contains("*"))
{
string[] parts = line.Split('*');
if (int.TryParse(parts[1], out int multiplier) && (multiplier &
{
int baseNum = int.Parse(parts[0]);
int result = baseNum << Log2(multiplier); // Replace multip
return result.ToString();
}
}
121 %
No issues found
Ln: 42
Ch: 49
SPC
CRLF
Output
Show output from:
Error List
Output
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'OptimizationLayer' (1 of 1 project)
OptimizationLayer
Dependencies
Form1.cs
Form1.Designer.cs
Form1.resx
Program.cs
Solution Explorer
Git Changes
Properties
Ready
Add to Source Control
Select Repository
2:38 pm
28/12/2023